elektrorevue
ISSN 1213 - 1539

Rok / Year:	Svazek / Volume:	Číslo / Number:	Jazyk / Language
2018	19	6	CZ

Autentizace IoT zařízení protokolem OAuth2 nebo HTTP(S) pro ukládání Big Data přes REST API

Autentization of IoT devices using OAuth2 protocol or HTTP(S) for storing Big Data with REST API

Pavel Šeda, Pavel Mašek, Jiří Hošek

xsedap01@stud.feec.vutbr.cz, masekpavel@vutbr.cz, hosek@feec.vutbr.cz

Faculty of Electrical Engineering and Communication Technologies, Brno University of Technology

DOI: -

Abstract: Currently, the importance of IoT applications in the market is rapidly growing. Depending on this, the importance of proper storage of IoT applications increases, where it is advisable to send data to the server or to receive instructions from it. It is therefore advisable to deal with Big Data and IoT device server authentication, so in this thesis are compared OAuth2 and HTTP(S) protocols for authentication of IoT device. The output of this article is a comparison of the speed of data storage when sending data as JSON or XML messages as well as the authentication method via HTTP Digest or OAuth2 protocol.

elektrorevue
ISSN 1213 - 1539

Autentizace IoT zařízení protokolem OAuth2 nebo HTTP(S) pro ukládání Big Data přes REST API

Pavel Seda, Pavel Masek, Jiri Hosek

Fakulta elektrotechniky a komunikačních technologií VUT v Brně
Email: xsedap01@stud.feec.vutbr.cz, masekpavel@vutbr.cz, hosek@feec.vutbr.cz

Abstrakt *– V dnešní době velmi rychle roste význam IoT aplikací na trhu. V závislosti na tom se zvyšuje význam vhodného ukládání dat z IoT aplikací, u kterých je vhodné posílat data na server popřípadě od něj dostávat instrukce. Je tedy vhodné zabývat se problematikou Big Data a také autentizací komunikace IoT zařízení se serverem, proto jsou v této práci modelovány a porovnány autentizace IoT zařízení protokoly OAuth2 nebo HTTP(S), které patří mezi základní 2 standardy pro autentizaci. Výstupem článku je srovnání rychlosti uložení dat při zasílání zpráv ve formátu JSON nebo XML a rovněž při způsobu autentizace přes HTTP Digest nebo OAuth2.*

1 Úvod

IoT (Internet of Things) a BigData v dnešní době představují velkou výzvu pro poskytovatele služeb v IoT průmyslových aplikacích. Tato výzva je dána zejména velkým množstvím dat, které putují po přenosových linkách a jejich vhodným ukládáním [1]. Nevhodná volba formátu ukládání dat nebo nevhodná volba databázového systému může výrazně prodražit běh takových aplikací.

Problematika BigData se zabývá datovými sklady, které jsou velké a tak komplexní, že je nevhodné je ukládat skrze standardní relační databázové systémy a často se proto volí pro ukládání a práci s takovými daty databáze nerelačního typu tzv. NoSQL databáze. Tyto NoSQL databáze jsou často ukládány ve formátu JSON (JavaScript Object Notation) a umožňují efektivnější dotazování, vizualizaci, sdílení a vyhledávání v těchto rozsáhlých datových uložištích.

Jako rozhraní pro komunikaci mezi IoT zařízením a daným typem databázového serveru se obvykle používání REST API (Representational State Transfer Application Interface), který přijímá žádosti zpravidla ve formátu JSON nebo XML a dotazuje se databázového serveru na potřebná data, která následně vrací pro IoT zařízení. Je žádoucí, aby komunikace IoT zařízení se serverem s REST API byla vhodně autentizována pro uchování určité úrovně bezpečnosti.

Přínosem této publikace jsou modely autentizace IoT zařízení se serverovou částí prostřednictvím protokolů OAuth2 a HTTP Digest. Dále je provedeno srovnání času nezbytného pro uložení většího množství dat při použití formátu JSON nebo XML a v neposlední řadě je ověřena časová složitost uložení dat při autentizaci HTTP Digest a OAuth2.

2 REST API

Jde o architekturu, která byla navržena v rámci doktorské práce Roye Fieldinga v roce 2000. Jde o softwarovou architekturu, která je určitým druhem reverzního inženýrství vzhledem k funkcím webu. Roy Fielding je spoluautor protokolu HTTP, což ovlivnilo tvorbu REST architektury, která pracuje obvykle přes protokol HTTP/HTTPS. REST API využívá metody HTTP jako jsou @GET, @PUT, @POST, @HEAD, @OPTIONS, @PATCH [6, 3].

2.1 Základní principy restu

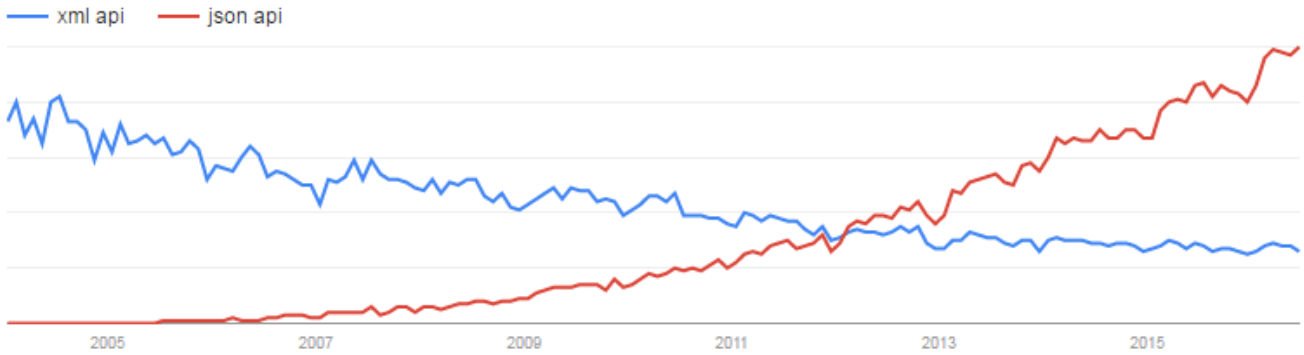
- Stav serveru: je určen na základě takzvaných resource (zdrojů dat, druh abstrakce), kde každý takovýto resource musí být jednoznačně identifikován pomocí URL (Uniform Resource Locator).
- Resource: může být reprezentován různými typy souborů jako je JSON, XML, PDF, SVG atd., kde IoT zařízení nepracuje přímo s resource, ale s jeho reprezentací.
- Operace: s resource je definován jednotný přístup v podobě čtyř operací čtení, zápisu, modifikace a mazání.
- HATEOS (Hypermedia as the Engine of Application State): vyjadřuje, že každý resource je určen pomocí URL a další stavy jsou získány pomocí odkazů.

2.2 Srovnání obvyklých reprezentací resource

Z obrázku 1 je vidět, že JSON API postupně nahrazuje XML. Tento trend je dán úsporností JSON reprezentace oproti XML. JSON reprezentace rovněž méně zatěžuje síťové linky, protože není potřeba přenášet tak velký objem dat. Rychlostí zpracování dosahuje JSON také lepších výsledků, protože je obvykle jednodušší na získávání potřebných dat.

Pro názornost je níže uveden příklad zápisu uživatele IoT zařízení ve formátu JSON [5]:

elektrorevue
ISSN 1213 - 1539



Obrázek 1: Srovnání využívání formátu XML a JSON [4].

```
{"uzivatele":[
    { "jmeno":"Pavel",
      "prijmeni":"Seda",
      "datum":"21.07.2017",
      "usleKM":"5",}
]}
```

a dále ve formátu XML.

```
<uzivatele>
    <uzivatel>
        <jmeno>Pavel</jmeno>
        <prijmeni>Seda</prijmeni>
        <datum>21.07.2017</datum>
        <usleKM>5</usleKM>
    </uzivatel>
<uzivatele>
```

Z ukázky výše je zřetelné, že formát JSON je úspornější. Z tohoto důvodu se obvykle používání pro ukládání velkého množství dat formát JSON (problematika Big Data). Je velmi vhodné přenášet po síti, co nejmenší možný objem dat, aby se síť příliš nezahlcovala a server nebyl příliš zatěžován výkonnostně náročným parsováním XML dokumentů.

2.3 Architektura REST API

Implementace serverové strany s využitím REST API je znázorněna na obrázku 2. Z obrázku je vidět, že nejde o protokol, ale architekturu, která vytváří další abstraktní úroveň pracující nad databází [6]. Tato abstraktní vrstva umožňuje zajistit další úroveň ochrany, protože nad databází nejde volat „destruktivní dotazy“ (pokud nejsou nadefinovány v REST API), které by mohly mít nepříznivý dopad na databázi. Příkladem takových dotazů může být následující SQL dotaz.

```
    SELECT * FROM users
```

Bez stanovení limitů na počet vrácených instancí z tabulky uživatelé (users), by databáze vrátila všechny uživatele, to by v případě, že tabulka uživatelé bude obsahovat miliony záznamů o uživateli IoT zařízení, mělo drtivý dopad na výkon databáze. Zatímco pokud bychom takový dotaz volali přes REST API, tak REST API už může být jednoduše nastavena, že při takovém dotazu se nastaví maximální limit vrácených instancí například 100.

3 OAuth2 a HTTP autentizace

Pro ověření autentičnosti v IoT se používají různé autentizační protokoly, které by měly počítat s poměrně nízkým hardwarovým vybavením IoT zařízení. Je potřeba počítat s co nejmenšími potřebnými úkony, aby jej zařízení bylo schopné v rozumném čase provést a zároveň autentizace neztratila na úrovni bezpečnosti.

V této sekci jsou představeny protokoly OAuth2 a HTTP pro autentizaci IoT zařízení, které se často používají i pro autentizaci standardních počítačových aplikací.

Mezi některé další protokoly pro autentizaci patří MQTT (Message Queuing Telemtry Transport), PAuthKey (Pervasive Authentication Protocol And Key Establishment), CoAP (The Constrained Application Protocol), DTLS (Datagram Transport Layer Security), HDM16 (Multidevice protokol) [9].

3.1 OAuth2

OAuth2 je protokol, který umožňuje aplikacím třetích stran např. Raspberry Pi 3, získat limitovaný přístup k REST službě (nebo HTTP službě obecně).

OAuth verze 2 byla vytvořena pro zjednodušení procesu autentizace a je široce rozšířena i mezi softwarovými giganty mezi, které se řadí Facebook a Google [7].

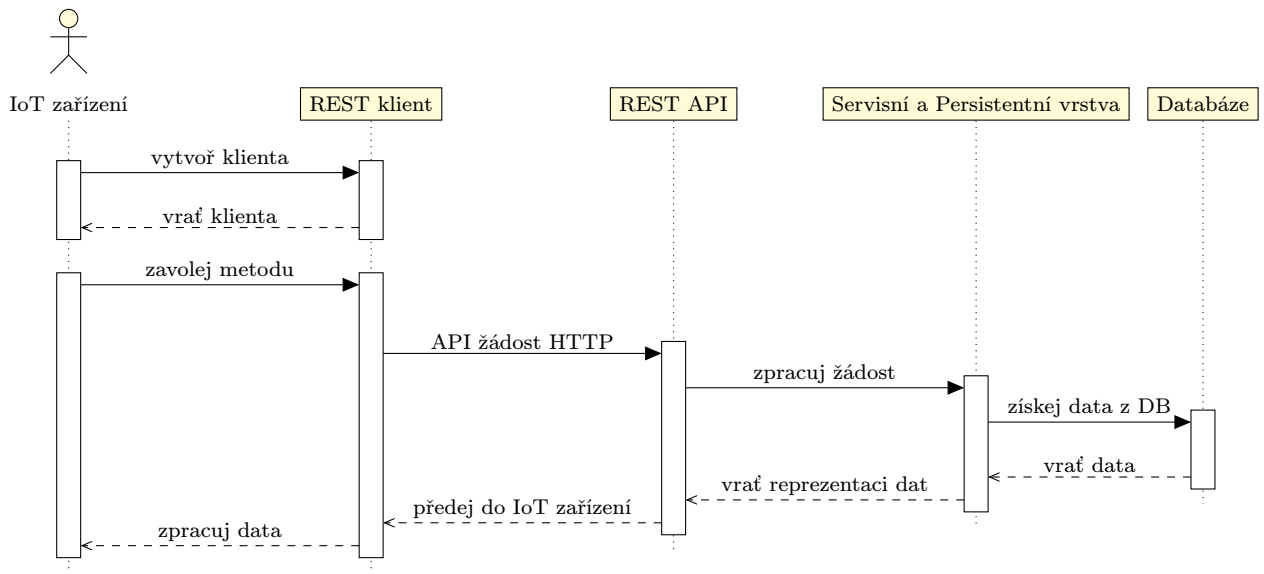
V autentizaci protokolem OAuth2 jsou definovány 4 role, které jsou podrobněji popsány v následující sekci.

3.1.1 Role v OAuth2

OAuth2 definuje 4 role, které spolu komunikují během autentizace zařízení. Průběh komunikace jednotlivých rolí je znázorněn na obrázku 3.

- IoT zařízení (Client): Zařízení, které se snaží získat přístup ke sdíleným datům. Může jít například

elektrorevue
ISSN 1213 - 1539



Obrázek 2: Architektura REST API na straně serveru.

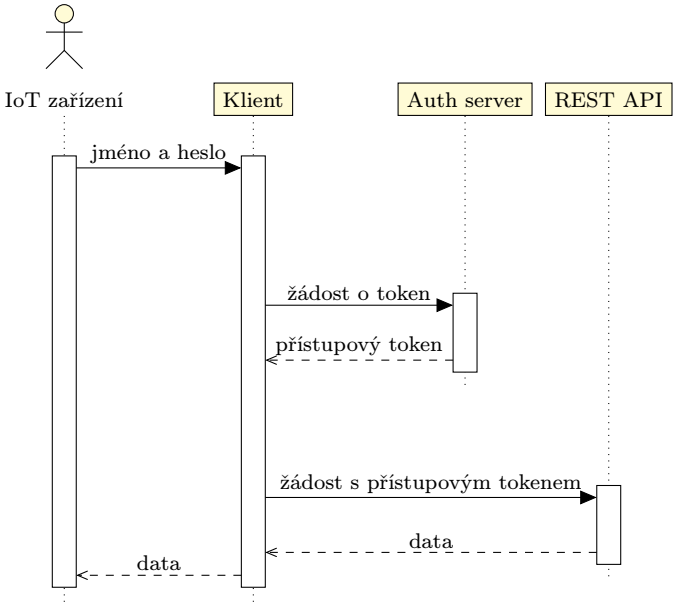
o Raspberry Pi 3.

- Klient (Client): Aplikace, která žádá o přístup server pro autentizaci, který mu vydává dočasný přístupový token na základě klientova id, přesměrované uri a typu odpovědního kódu.
- Server pro autorizaci (Authorization Server): Server, který zpracovává žádosti o token a vydává dočasně platné tokeny pro komunikaci s REST API. Server pro autorizaci obvykle bývá zakomponován.
- REST API (Resource Server/Resource Owner): Server nad chráněnými daty (může jít o databázi přímo nebo API pracující s databází.).

3.1.2 Tokeny v OAuth2

Tokeny jsou náhodně vygenerované řetězce serverem pro autentizaci. Jsou vydány v případě, že o ně klient zažádá. Existují 2 typy tokenů, které jsou popsány níže.

- Přístupový token (Access Token): Jde o token, který je poslán klientem jako parametr nebo v hlavičce HTTP dotazu na REST API server. Přístupový token má obvykle omezenou životnost, která je definována serverem pro autentizaci. Musí být při komunikaci a v uložišti důvěrný. Pouze strany, které by měly znát tento token jsou klient, server pro autentizaci a REST API. Klient by měl zajistit, že přístupový token není přístupný jiným aplikacím na stejném IoT zařízení. Přístupový token by rovněž měl být použitelný pouze přes HTTPS připojení, protože komunikace přes nezabezpečený kanál by znamenala jednoduché odposlechnutí tohoto tokenu.
- Token pro obnovu (Refresh Token): Token vydán serverem pro autentizaci, který na rozdíl od přístupového tokenu není použit při každé žádosti na REST API. Slouží k obnovení přístupového tokenu, kterým klient přistupuje k datům. Žádost o obnovení přístupového tokenu se posílá přibližně, když je přístupový token ve stavu 3/4 jeho životnosti [8].



Obrázek 3: Autentizace protokolem OAuth2.

3.1.3 Použití přístupového tokenu

Přístupový token může být zaslán několika způsoby na REST API server.

Zaslání tokenu přes autentizační hlavičku:

```
GET /profile HTTP/1.1
```

elektrorevue ISSN 1213 - 1539

```
Host: api.rest.iot
Authorization: Bearer ACCESS_TOKEN_STRING
```

Tento způsob je efektivní, ale není podporován všemi REST API servery.

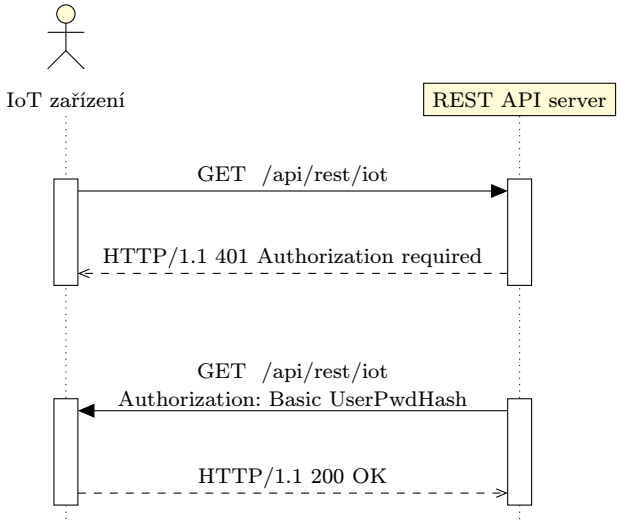
Poslání tokenu jako parameter dotazu:

```
GET ~/profile?access_token=ACCESS_TOKEN_STRING
```

Tento způsob není ideální vzhledem k tomu, že přístupový token může být nalezen v lozích REST API serveru. Nicméně vzhledem ke své snadné použitelnosti a plné podpoře serverů je to nejčastější způsob zaslání tokenu. Token se tímto způsobem dá zaslat buď HTTP metodou GET nebo HTTP metodou POST.

3.2 HTTP autentizace

HTTP narozdíl od OAuth2 autentizace nevyžaduje nutnost použití přístupových tokenů jako je vidět na obrázku 4.



Obrázek 4: Autentizace protokolem HTTP.

Počáteční žádost ze strany IoT zařízení je typicky anonymní žádost, neobsahující žádné autentizační informace. REST API server poté může indikovat, že anonymní žádost k určitému zdroji informace není povolen a je nutná autentizace případě autentizace s určitým oprávněním. Server zašle WWW-Authentication hlavičku, která obsahuje informaci o tom jaké autentizační schéma sever vyžaduje. V tabulce 1 jsou vidět některé možnosti autentizačních schémat [13].

Tabulka 1: Autentizace pomocí HTTP schémat.

Schéma	Popis
Anonymous	Anonymous je autentizace, kdy není zasíláno žádné autentizační informace. Představuje ekvivalent poskytnutí přístupu ke zdroji informace každému, kdo o ni zažádá.
Basic	Basic je autentizace, kdy zasíláme na server BASE64-encoded [10] řetězec kódovaný pomocí BASE64) řetězec, který obsahuje jméno a heslo z IoT zařízení. BASE64 není považován jako forma kódování, protože zjednodušeně představuje odesílání uživatelského jména a hesla v čistém textu. Pokud by zdroj měl být zabezpečen, tak je silně doporučováno použít jiné autentizační schéma než HTTP Basic.
Digest	Digest autentizace je určena k nahrazení Basic autentizace. Jde o schéma „challenge-response“, kde server pošle řetězec náhodných dat nazývaný „nonce“ IoT zařízení jako „challenge“. Klient odpoví hashem, který obsahuje uživatelské jméno a heslo a „nonce“ mezi dalšími údaji. Komplexnost této výměny a datové hashování dělají z Digest autentizace výrazně složitější typ schématu pro útočníka na odposlech nebo znovupoužití uživatelského jména a hesla [12, 14].

3.2.1 Výběr vhodného typu schématu

Při výběru vhodné typu schématu pro REST API autentizaci je vhodné zvážit následující body:

- zda je nutné mít všechny data chráněna autentizací, protože HTTP autentizace vyžaduje přenos většího množství dat a může zpomalovat interakci s IoT zařízením. Proto je vhodné povolit přístup k některým

elektrorevue ISSN 1213 - 1539

datům všem a tím zrychlit komunikaci mezi IoT zařízením a serverem.

- Pokud patřičná data musí být zabezpečena, tak je vhodné volit spíše autentizaci Digest místo Basic, protože zátěž na provozních linkách je srovnatelná a navíc je autentizace pomocí Digest výrazně bezpečnější. Implementačně jsou oba způsoby srovnatelně obtížné [13, 12].

4 Vytvoření benchmarků

Na základě vytvořených modelů komunikace IoT zařízení se vzdálenou databází viz model 2, procesů autentizace pomocí HTTP, konkrétně dle schématu HTTP Basic a OAuth2 protokolu, byla implementována serverová část aplikace.

V této sekci jsou představeny výsledky z dvou vytvořených benchmarků, kde první porovnává rychlost zpracování formátu JSON a XML a druhý rychlost uložení při použití autentizace HTTP Digest nebo OAuth2.

4.1 Porovnání zpracování XML a JSON formátu

Bylo provedeno srovnání souborových formátů XML a JSON pro ukládání dat přes REST API s autentizací dle schématu HTTP Digest do vzdálené databáze. Byl vytvořen benchmark, na kterém bylo testováno uložení 500 000 záznamů ve formátu JSON:

```
{"devices":[
    { "device_id":"1",
      "longitude":"49.1450",
      "latitude":"17.9894",
      "battery":"20%"
    }
]}
```

a ve formátu XML.

```
<devices>
    <device>
        <device_id>1</device_id>
        <longitude>49.1450</longitude>
        <latitude>17.9894</latitude>
        <battery>20%</battery>
    </device>
<devices>
```

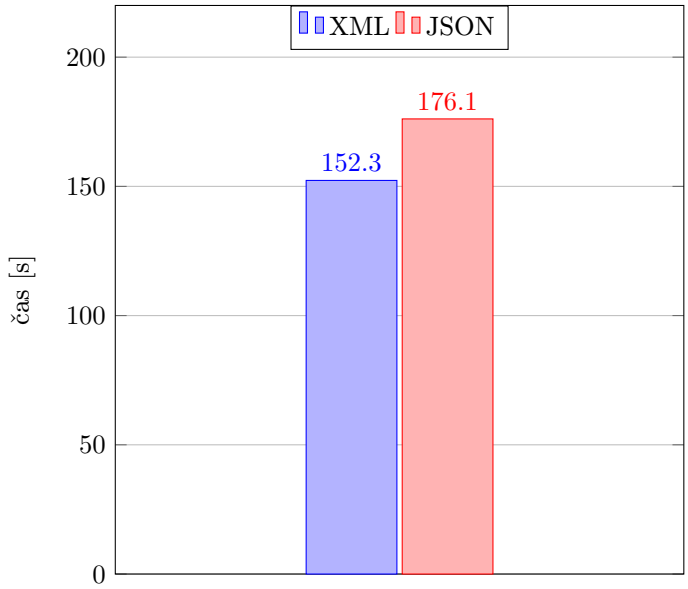
Testy byly pro zvýšení přesnosti výsledků opakovaně spuštěny 150krát a následně zprůměrována (vyšší počet testů není nezbytný, protože další měření již výsledné hodnoty měnily zanedbatelně). Jako parser příchozích zpráv na straně serveru byl v případě formátu JSON použit Jackson Databind, který umí automaticky serializovat formát JSON. Pro použití tohoto mapperu v Javě při použití systému Maven [15] stačí vložit závislost na knihovnu dle následujícího kódu.

```
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.9.1</version>
</dependency>
```

Pro parsování XML zpráv byl použit Jackson XML, což je rozšíření Jackson JSON na čtení a zápis XML dat. V Javě při použití systému Maven je nutné vložit následující závislost.

```
<dependency>
    <groupId>com.fasterxml.jackson.dataformat
    </groupId>
    <artifactId>jackson-dataformat-xml
    </artifactId>
    <version>2.9.2</version>
</dependency>
```

Na obrázku 5 je vidět, že zpracování dat ve formátu JSON je rychlejší než zpracování dat ve formátu XML. Je to z důvodu úspornosti JSON formátu.

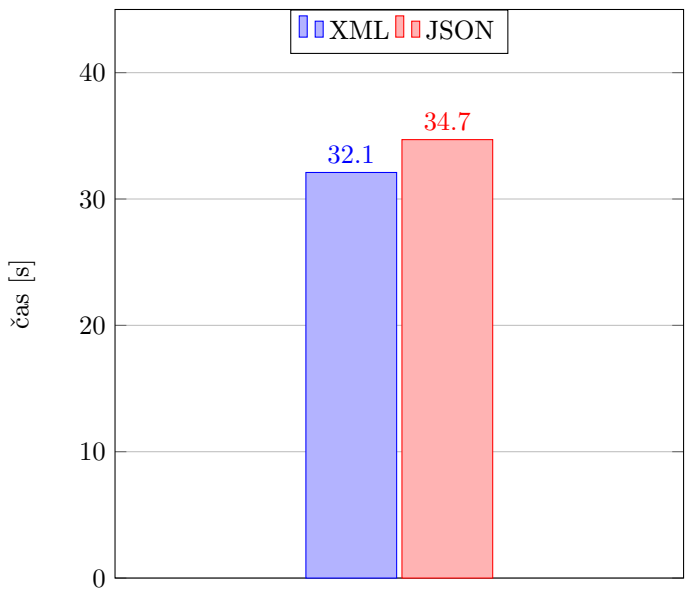


Obrázek 5: Srovnání času pro uložení 500 000 záznamů ve formátu XML a JSON dat.

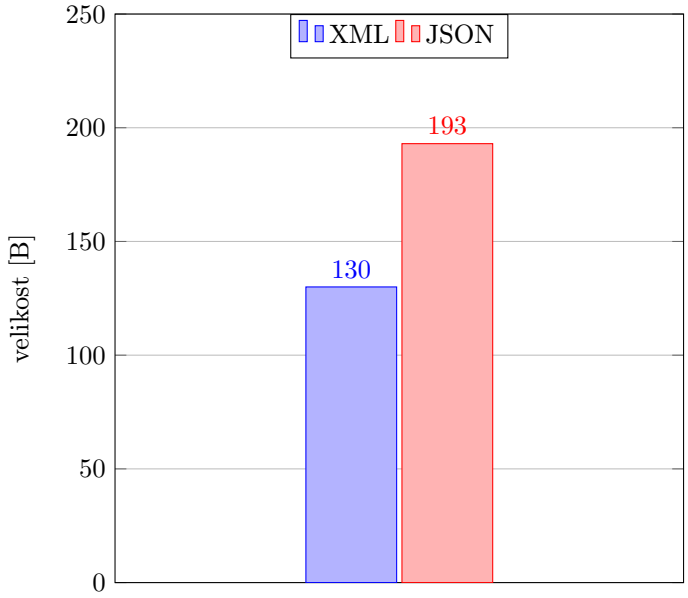
Pro srovnání bylo provedeno další měření pro uložení 100 000 záznamů. Výsledek z tohoto měření je vidět na Obr. 6. Na obrázku lze vidět, že se oproti uložení 500 000 záznamů času zpracování zpráv ve formátu XML poměrově zhoršil. To je pravděpodobně dáno tím, že u většího množství dat se více projeví gzip komprese, která je velmi účinná právě u formátu XML [11].

Pro úplnost je na obrázku 7 uvedena i velikost těchto zpráv v bajtech.

elektrorevue
ISSN 1213 - 1539



Obrázek 6: Srovnání času pro uložení 100 000 záznamů ve formě XML a JSON dat.

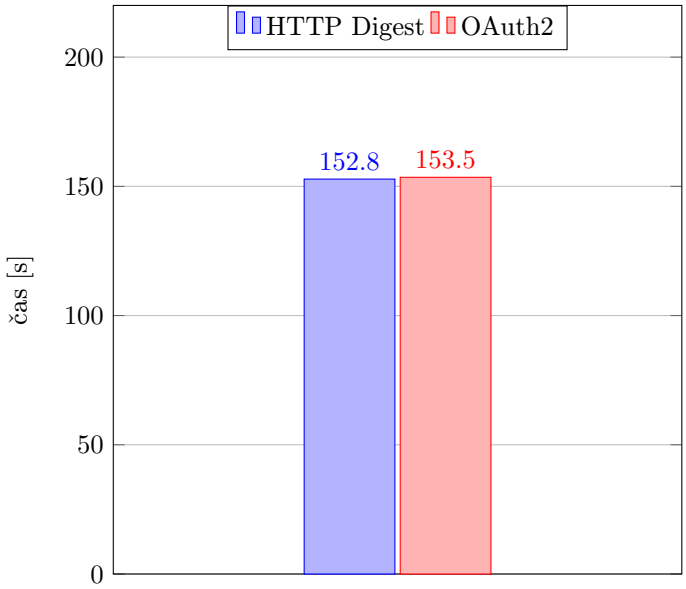


Obrázek 7: Srovnání velikosti zasílaných zpráv ve formát JSON a XML.

4.2 Porovnání zpracování dat při HTTP Digest a OAuth2 autentizaci

Srovnání autentizace HTTP Digest a OAuth2 bylo provedeno na vzorku dat ve formátu JSON. Bylo opět zaslání 500 000 záznamů o zařízení. Z průběhů autentizace viz 3 a 4 je zřejmé, že výsledky by se neměly příliš lišit. To dokazuje i výsledek z tohoto benchmarku, který je vidět na obrázku obrázku 8. Rozdíl mezi rychlostí uložení záznamů při použití autentizace HTTP Digest a při použití autentizace OAuth2 se liší o 1 sekundu. Tento výsledek je dán tím, že po vyžádání token autentizačním serverem se už IoT zařízení dotazuje autentizačního serveru o token pouze pokud je token v 3/4 jeho expirace.

Výsledek tohoto benchmarku ukazuje, že použití autentizace OAuth2 výrazně nezvyšuje nezbytný čas pro uložení záznamů do databáze.



Obrázek 8: Srovnání rychlosti zpracování dat při autentizaci HTTP Digest a OAuth2.

5 Závěr

V tomto článku byly namodelovány procesy autentizace pomocí HTTP, konkrétně dle schématu HTTP Basic a OAuth2 protokolu, včetně způsobů zasílání potřebných údajů jako je uživatelské jméno a heslo v zašifrované podobě nebo pomocí přístupových tokenů.

Byla zhodnocena vhodnost použití těchto bezpečnostních aspektů u vybraných typů aplikací, kde u aplikací, které nemají příliš velký důraz na bezpečnost je v současné době dostatečně vyhovující autentizace pomocí HTTP schématu Digest, který komplexním způsobem šifruje uživatelské jméno a heslo. Je rovněž vhodné pro některé typy operací ponechat přístup zcela bez autentizace. Tím se zmenší objem přenesených dat, to je zcela zásadní v problematice Big Data, která je úzce svázána s ukládáním velkého množství dat z IoT zařízení. Při velkém provozu je žádoucí, co nejméně zatěžovat přenosové linky.

Tato problematika je vzhledem k progresivnímu rozvoji IoT zařízení v průmyslových aplikací velmi aktuální, protože je vyšší potřeba ukládat data ve vzdálených databázích. Jako efektivní způsob pro komunikaci mezi IoT zařízením a serverem a ukládání dat ve vzdálených databázích se jeví architektura REST API, která je poměrně jednoduchá na implementaci, jak na straně serveru, tak ze strany

elektrorevue
ISSN 1213 - 1539

IoT zařízení. IoT zařízením tak stačí pouze posílat HTTP dotazy s tělem reprezentovaným ve formě JSON dokumentu nebo pouze skrze parametry zadané adresy. Oproti protokolu SOAP, který využívá pouze XML reprezentaci pro přenos zpráv také méně zatěžuje přenosové linky, protože JSON dokumenty jsou obsahově menší.

Poděkování

Výzkum popsaný v tomto článku byl financován z Národního programu udržitelnosti, grant č. LO1401. Pro výzkum byla použita infrastruktura Centra SIX. Výzkum popsaný v tomto článku byl podpořen grantem VI20162018007 Ministerstva vnitra, Programem bezpečnostního výzkumu České republiky 2015-2020.

Literatura

[1] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update [online]. [cit. 2017-10-17]. Dostupné z: `https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html`

[2] MASSÉ, Mark. *REST API design rulebook*. Sebastopol, CA: O'Reilly, 2012. ISBN 9781449310509.

[3] ŠEDA, Pavel. *Mobilní aplikace pro subjektivní měření kvality zážitku streamovaného videa*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2017. 87 s. Vedoucí diplomové práce Ing. Dominik Kováč.

[4] KÅGSTRÖM, Jon. *uClassify* [online]. [cit. 29.9.2017]. Dostupné z: `http://blog.uclassify.com/json-rest-api/`

[5] HOLUBOVÁ, Irena, Jiří KOSEK, Karel MINAŘÍK a David NOVÁK. *Big Data a NoSQL databáze*. Praha: Grada, 2015. Profesionál. ISBN 978-80-247-5466-6.

[6] MASSÉ, Mark. *REST API design rulebook*. Sebastopol, CA: O'Reilly, 2012. ISBN 9781449310509.

[7] LEBLANC, Jonathan a Tim MESSERSCHMIDT. *Identity and data security for web development: best practices*. Boston: O'Reilly Media, 2016. ISBN 978-1491937013.

[8] RICHER, Justin, Antonio SANSO a Ian GLAZER. *OAuth 2 in action*. Shelter Island, NY: Manning Publications, 2017. ISBN 9781617293276.

[9] DRÁPELA, Roman. *Autentizace v IoT*. bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2017. 66 s. Vedoucí práce byl Ing. Petr Dzurenda

[10] WANG, Jie a Zachary A. KISSEL. *Introduction to Network Security* [online]. Singapore: John Wiley Sons Singapore Pte., 2015 [cit. 2017-10-16]. ISBN 9781119113102.

[11] YE FENG. *XML document compression based on BWT algorithm*. In: 2010 Second Pacific-Asia Conference on Circuits, Communications and System [online]. IEEE, 2010, s. 113-116 [cit. 2017-12-17]. DOI: 10.1109/PACCS.2010.5627004. ISBN 978-1-4244-7969-6. Dostupné z: `http://ieeexplore.ieee.org/document/5627004/`

[12] REITAN, Erik a kol. *Microsoft Documentation* [online]. [cit. 29.9.2017]. Dostupné z: `https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/understanding-http-authentication`

[13] GOURLEY, David. a Brian. TOTTY. *HTTP: the definitive guide*. Sebastopol, CA: O'Reilly, 2002. ISBN 978-1565925090.

[14] FIELDING, Roy a Jim GETTYS. *Hypertext Transfer Protocol – HTTP/1.1* [online]. United States, 1999 [cit. 2017-02-01]. Dostupné z: `https://tools.ietf.org/html/rfc2616`

[15] VARANASI, Balaji a Sudha BELIDA. *Introducing Maven*. Berkeley, CA: Apress, 2014. Expert's voice in Java. ISBN 978-1484208427.